SANDIA REPORT

SAND2005-0935
Unlimited Release
Printed March 2005

EMPHASIS/Nevada UTDEM User Guide Version 1.0

C. David Turner, Michael F. Pasik, David B. Seidel

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

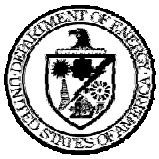
Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online



SAND2005-0935
Unlimited Release
Printed March 2005

EMPHASIS/Nevada UTDEM User Guide Version 1.0

C. David Turner, Michael F. Pasik, David B. Seidel

Electromagnetics and Plasma Physics Analysis

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-1152

Abstract

The Unstructured Time-Domain ElectroMagnetics (UTDEM) portion of the EMPHASIS suite solves Maxwell's equations using finite-element techniques on unstructured meshes. This document provides user-specific information to facilitate the use of the code for applications of interest.

Acknowledgement

The authors would like to thank all of those individuals who have helped to bring EMPHASIS/Nevada to the point it is today, including Bill Bohnhoff, Rich Drake, and all of the NEVADA code team.

Contents

List of Figures

Intentionally Left Blank

Introduction

EMPHASIS/Nevada Unstructured Time-Domain ElectroMagnetics (UTDEM) is a general-purpose code for solving Maxwell's equations on arbitrary, unstructured tetrahedral meshes. The geometries and the meshes thereof are limited only by the patience of the user in meshing and by the available computing resources for the solution. UTDEM solves Maxwell's equations using finite-element method (FEM) techniques on tetrahedral elements using vector, edge-conforming basis functions [1].

EMPHASIS/Nevada Unstructured Time-Domain ElectroMagnetic Particle-In-Cell (UTDEM PIC) is a superset of the capabilities found in UTDEM. It adds the capability to simulate systems in which the effects of free charge are important and need to be treated in a self-consistent manner. This is done by integrating the equations of motion for macroparticles (a macroparticle is an object that represents a large number of real physical particles, all with the same position and momentum) being accelerated by the electromagnetic forces upon the particle (Lorentz force). The motion of these particles results in a current, which is a source for the fields in Maxwell's equations.

UTDEM Simulation Process

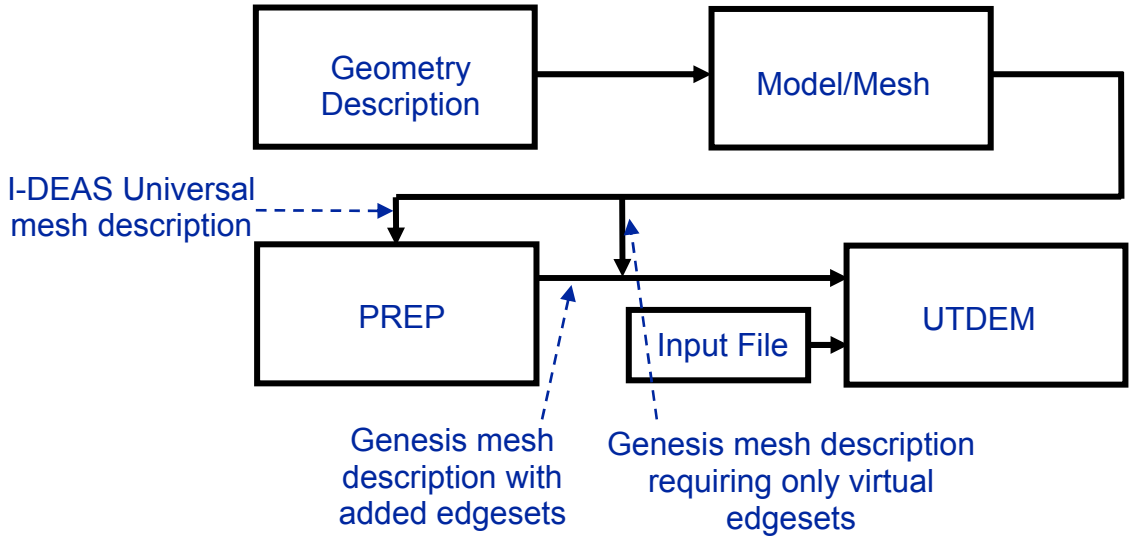


Figure 1. UTDEM Simulation Process.

The UTDEM simulation process is shown in Fig. 1. The geometry of interest must first be modeled and meshed using appropriate software. I-DEAS is one option that provides both, but successful meshes have been generated using CUBIT and ICEM CFD Tetra as well. Regardless of how the mesh is generated it must ultimately be written or converted to EXODUSII [2] format. CUBIT and ICEM do this directly as does I-DEAS with the

addition of a special 3rd party feature. However, certain features of UTDEM require sets of nodes (nodesets), edges (edgesets), or faces (sidesets) to be defined. In particular, edgesets are not a part of the EXODUSII standard. The creation of these "edgesets" along with nodesets and sidesets are handled along with the conversion from I-DEAS universal format to EXODUSII format by the preprocessing step PREP [3]. Use of virtual edgesets (described later) avoids the requirement to process edgesets through PREP. Further discussion of the use of PREP is found in Appendix I.

The actual input required for UTDEM consists of only two files, the Genesis file containing the mesh and the problem-specific input file. "Genesis" normally describes a mesh description file in EXODUSII format containing mesh only, no data.

A critical aspect of the mesh for UTDEM is that boundary conditions are described by EXODUSII sidesets, edgesets, or nodesets. These requirements will be described later as each feature is covered in detail. The tetrahedral elements should also be reasonably well shaped.

UTDEM results are available in the form of observer time histories and plot dumps of specified variables in an EXODUSII file. These results can be displayed with most any simple plotting package in the case of time histories and by post-processing tools which can import EXODUSII, such as EnSight or Blot, in the case of plot dumps.

UTDEM Input File and Keywords

The UTDEM input file uses the standard NEVADA input file format, which includes keywords for debugging, physics type, solver control, output control, and more. Details of all of these except for the specific physics can be found in the ALEGRA users guide [4]. A complete input file for one of the UTDEM regression problems is given in Appendix II.

```
UNSTRUCTURED TD ELECTROMAGNETICS
  formulation, helmholtz
  aztec set, 0
  abc bc, sideset 4
  pec bc, sideset 2
  observer, nodeset 28
  observer, nodeset 29
  source, nodeset 31, function 1
  slot observer, nodeset 19
  slot, edgeset 123, aztec_set 1, width 0.00001, depth 0.0, int_mat 1,
ext_mat 2
  slot observer, nodeset 20
  slot, edgeset 124, aztec_set 2, width 0.00005, depth 0.0, int_mat 1,
ext_mat 2
END
```

Figure 2. Typical UTDEM physics keywords.

The format for specifying UTDEM physics and associated keywords is shown in Fig. 2. The keyword "**unstructured td electromagnetics"** specifies to the Nevada framework that the UTDEM physics model should be used for the simulation. Most of the remaining keywords are specific to UTDEM and are described below along with many others. Case is ignored in the input file. It is important to note that lines are limited to 160 characters or less. Keywords can be abbreviated to stay below this limit. Additionally, when a floating-point value is required, the decimal point needs to be included (i.e., 0. not simply 0).

Solver formulation keyword

formulation

Specifies the UTDEM solver formulation for this simulation

formulation, *type*

Options for *type* are:

helmholtz

Utilize the unconditionally stable, 2nd order formulation (recommended)

curlcurl

Utilize the conditionally stable, coupled 1st order formulation (not fully implemented)

Boundary condition keywords

pec bc

Specifies a perfect electric conductor boundary condition

pec bc, sideset *int*

The electric fields tangential to the surface described by the sideset having id **sideset will be set to zero**. This sideset id must exist in the genesis file or Nevada will complain. Multiple pec bc's may exist in the same simulation.

As with all sidesets, multiple disjoint surfaces may exist in the same sideset. PEC surfaces may be grouped into different sidesets for purposes of visualization. (NOTE: it might make sense to have a section on side/edge/node sets since many of the above comments will be repeated for each BC/feature).

abc bc

Specifies an absorbing boundary condition

abc bc, sideset *int*

A 1st order absorbing boundary condition will be applied over the surface described by the sideset having id **sideset**. This sideset id must exist in the genesis file or Nevada will complain. Multiple abc bc's may exist in the same simulation.

An absorbing boundary condition is typically specified in conjunction with a port boundary condition when the port is located on an exterior surface.

ibc bc

Specifies an impedance boundary condition

ibc bc, sideset *int***, impedance** *real*

A 1st order surface impedance boundary condition with surface impedance value **impedance** will be applied over the surface described by the sideset having id **sideset**. Presently, this "impedance" must be real, a surface resistance only. This sideset id must exist in the genesis file or Nevada will complain. Multiple ibc bc's may exist in the same simulation.

Boundary condition precedence is presently specified in the code as follows: PEC->IBC->ABC, i.e., an edge specified as both PEC and IBC or ABC will be PEC. An edge specified as both IBC and ABC will be IBC.

Application of the Perfect Magnetic Conductor (**PMC**) or mirror boundary condition is trivial since it is the natural boundary condition for the edge-conforming FEM formulation. The user simply leaves the mesh "free" on the surface where the PMC is to be applied.

Virtual edgeset keyword

path

Specifies a virtual edgeset to be created along an arbitrary user-defined path through the geometry, independent of mesh topology.

path, edgeset *int*, **point**, **x**=*real* **y**=*real* **z**=*real* **point, x**=*real* **y**=*real* **z**=*real* **point, . . .**

The best-fitting path of existing edges to match the desired line segment(s) entered will be found. The user is responsible for picking a unique **edgeset** id. If an id is chosen which exists in the Genesis file, a warning is issued and the virtual edgeset is not used. If a specified segment endpoint cannot

be reached within the default tolerance (half the local minimum edge length) a warning will be issued giving the actual endpoint used.

These virtual edgesets may be used anywhere a mesh-defined edgeset can be used. The exodus edgesets keyword should not be applied to these edgesets.

Observer keywords

observer

The following syntax specifies the simplest type of single-edge, electric-field-projection observer:

observer, nodeset *int*

The edge will be located which is defined by the nodeset having id **nodeset**. This nodeset MUST contain only 2 nodes. Presently, UTDEM will not issue an error if it does not, the observer will simply not be created. This will be fixed in a future version. Multiple observers of this type may exist in the same simulation.

The observer time history will be written to a file with a generated name in the following format: *problem_name***.obs***nodeset_id***.***proc_id***.dat**, where *nodeset_id* is the **nodeset** id parameter from the **observer** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, each processor who has the observer is synchronized to the correct value from the owner and writes a separate observer time-history file. All of these files are identical at the end of the simulation.

The observer time history will be written to the hisplt database *problem*_name.**his** with the name: **OBS-***nodeset_id*.

slot voltage observer

The following syntax specifies the slot voltage observer:

slot voltage observer, nodeset *int*, [**direction x** *real* **y** *real* **z** *real*]

The slot voltage at the node defined by the nodeset having id **nodeset** will be monitored. This nodeset MUST contain only 1 node. An error will be issued if it contains more than 1 node. If the nodeset contains 0 nodes, the observer will simply not be created. Multiple slot observers may exist in the same simulation.

The assumed positive direction of the voltage (actually "magnetic current") can be specified by the optional parameter **direction**. The direction here is that *along* the slot, normal to the slot "gap" where the voltage would be measured across. If not specified, the natural direction (code internally assumed direction) will be output. This can cause inversion of the results in parallel depending on the number of processors. Consequently, it is recommended that the direction be specified. If the specified direction is not very close to the assumed direction, a warning is issued.

The observer time history will be written to a file with a generated name in the following format: *problem_name***.slotobs***nodeset_id***.***proc_id***.dat**, where *nodeset_id* is the **nodeset** id parameter from the **observer** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, each processor who has the observer is synchronized to the correct value from the owner and writes a separate observer time-history file. All of these files are identical at the end of the simulation.

The observer time history will be written to the hisplt database *problem*_name.**his** with the name: **SLOTVOLOBS-***nodeset_id*.

wire current observer

The following syntax specifies the wire current observer:

wire current observer, nodeset *int*, [**direction x** *real* **y** *real* **z** *real*]

The wire current at the node defined by the nodeset having id **nodeset** will be monitored. This nodeset MUST contain only 1 node. An error will be issued if it contains more than 1 node. If the nodeset contains 0 nodes, the observer will simply not be created. Multiple wire observers may exist in the same simulation.

The assumed positive direction of the current can be specified by the optional parameter **direction**. If not specified, the natural direction (code internally assumed direction) will be output. This can cause inversion of the results in parallel depending on the number of processors. Consequently, it is recommended that the direction be specified. If the specified direction is not very close to the assumed direction, a warning is issued.

The observer time history will be written to a file with a generated name in the following format: *problem_name***.wireobs***nodeset_id***.***proc_id***.dat**, where *nodeset_id* is the **nodeset** id parameter from the **observer** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, each processor who has the observer is synchronized

to the correct value from the owner and writes a separate observer time-history file. All of these files are identical at the end of the simulation.

The observer time history will be written to the hisplt database *problem*_name.**his** with the name: **WIRECUROBS-***nodeset_id*.

e line integral

The following syntax specifies an electric-field line-integral observer, $\int \overline{E} \bullet dl$:

e line integral, edgeset *int*, **direction x** *real* **y** *real* **z** *real*

The electric-field projections on the edges in the edgeset are simply summed. The vector **direction** is used to determine the direction of the integral and therefore determines the signs of all the individual edge contributions to the integral by taking the dot product of the vector edge direction with the **direction** vector. Generally, the mesh should be designed such that the edges are aligned in the correct direction, but this is not required. Multiple observers of this type may exist in the same simulation.

The observer time history will be written to a file with a generated name in the following format: *problem_name***.elinint***edgeset_id***.***proc_id***.dat**, where *edgeset_id* is the **edgeset** id parameter from the **observer** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, each processor who has a portion of the observer is synchronized to the correct value from the owner and writes a separate observer time-history file. All of these files are identical at the end of the simulation.

The observer time history will be written to the hisplt database *problem*_name.**his** with the name: **E*DL-***edgeset_id*.

h line integral

The following syntax specifies a magnetic-field line-integral observer, $\oint \overline{H} \bullet dl$:

h line integral, edgeset *int*

In this case, a direction vector is not required but the **user is responsible** for creating the mesh such that the beam elements defining the edgeset are oriented in the correct direction around the integration loop. An average magnetic field is computed for each edge in the edgeset by computing the

magnetic field at the center of each element connected to the edge. This vector magnetic field is then dotted with the edge direction and the "sense" of the edge, which is stored with the edgeset and is derived from the beam elements in the original mesh defining the edgeset. These dot products on the edges in the edgeset are then summed. Multiple hlinints may exist in the same simulation.

The observer time history will be written to a file with a generated name in the following format: *problem_name*.**hlinint***edgeset_id*.*proc_id*.**dat**, where *edgeset_id* is the **edgeset** id parameter from the **hlinint** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, each processor who has a portion of the observer is synchronized to the correct value from the owner and writes a separate observer time-history file. All of these files are identical at the end of the simulation.

The observer time history will be written to the hisplt database *problem*_name.**his** with the name: **H*DL-***edgeset_id*.

surface current

The following syntax enables the calculation of a surface current $n \times \overline{H}$ on the nodes of the specified conductor boundary sideset(s)

surface current, sideset *int* [*int* ...]

The surface current is computed from the magnetic field which itself is derived from the electric field. As a result, the magnetic field value used in the surface current calculation is constant over each element. An average normal is computed at each node from the normal of the element faces that contain the sideset node. To enable output of the computed surface current values, be sure to include SURFACE_CURRENT_DEN on the list of requested plot variables. Because some post-processing tools may have problems with the length of the variable name the following is suggested

```
plot variable
  surface_current_den, as "js"
end
```

The surface current will be output at all nodes in the simulation but will only have non-zero values on the specified sidesets.

Source keywords

source

The source keyword may be used in three different modes depending on the type of boundary set (nodeset, edgeset, or sideset) specified along with it.

1. The following syntax specifies the simplest type of single-edge, electric-field-projection Dirichlet source:

source, nodeset *int***, function** *int***, scale** *real* **shift** *real*

The edge will be located which is defined by the nodeset having id **nodeset**. This nodeset MUST contain only 2 nodes. Presently, UTDEM will issue an error if it has > 2 nodes. If it has 0 or 1 nodes, possibly due to parallel decomposition, the source will simply not be created. Multiple sources of this type may exist in the same simulation.

The keyword **function** *int* specifies a source time history defined by a Nevada keyword function definition described later. The function is scaled by **scale** and is shifted in time by **shift**.

The source time history will be written to a file with a generated name in the following format: *problem_name***.src***nodeset_id.proc_id***.dat**, where *nodeset_id* is the **nodeset** id parameter from the **source** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, each processor who has the source is synchronized to the correct value from the owner and writes a separate source time-history file. All of these files are identical at the end of the simulation.

The source time history will be written to the hisplt database *problem*_name.**his** with the name: **SRC-***nodeset_id*.

2. The following syntax specifies the next simplest electric-field-projection Dirichlet source--a multi-edge, linear, distributed source:

source, edgeset *int***, function** *int***, scale** *real* **shift** *real*, **direction**, **x** *real* **y** *real* **z** *real*, **length** *real*

The edges will be located which are defined by the edgeset having id **edgeset**. The source will be applied polarized along the direction vector **direction** and scaled to provide the desired magnitude when integrated over the length **length.** The scale factor for each edge in the source is $\overline{edge\_length} \bullet$ **direction** / **length**, where $\overline{edge\_length}$ is the vector edge length (not normalized). For this reason, this source makes sense only if

the edges are linear and along a Cartesian axis. Multiple sources of this type may exist in the same simulation.

The source time history will be written to a file with a generated name in the following format: *problem_name***.linsrc***edgeset_id.proc_id***.dat**, where *edgeset_id* is the **edgeset** id parameter from the **source** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, each processor who has the source is synchronized to the correct value from the owner and writes a separate source time-history file. All of these files are identical at the end of the simulation.

The source time history will be written to the hisplt database *problem*_name.**his** with the name: **LINEARSRC-***edgeset_id*.

3. The following syntax specifies the final type of electric-field-projection Dirichlet source--a multi-edge, belt-distributed source:

source, sideset *int***, function** *int***, scale** *real* **shift** *real,* **direction**, **x** *real* **y** *real* **z** *real*, **length** *real*

This source is applied over a surface and can, for example, be used to create a delta-gap source on a coax center conductor. The edges will be located which are defined by the sideset having id sideset. The source will be applied polarized along the direction vector **direction** and scaled to provide the desired magnitude when integrated over the length **length.** The scale factor for each edge in the source is $\overline{edge\_length} \bullet$ **direction** / **length**, where $\overline{edge\_length}$ is the vector edge length (not normalized). Multiple sources of this type may exist in the same simulation.

The source time history will be written to a file with a generated name in the following format: *problem_name***.dstsrc***sideset_id.proc_id***.dat**, where *sideset_id* is the **sideset** id parameter from the **source** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, each processor who has the source is synchronized to the correct value from the owner and writes a separate source time-history file. All of these files are identical at the end of the simulation.

The source time history will be written to the hisplt database *problem*_name.**his** with the name: **DISTSRC-***sideset_id*.

port source

The following syntax specifies a port source:

port source, coaxial, sideset *int***, function** *int***, scale** *real* **shift** *real***, center, x** *real***, y** *real***, z** *real*

or
port source, parallel plate, sideset *int*, **function** *int*, **scale** *real* **shift** *real*, **direction, x** *real* **y** *real* **z** *real*, **separ** *real*
or
port source, field dist, sideset *int*, **function** *int*, **scale** *real* **shift** *real*

Port sources are soft (non-Dirichlet) sources used to drive specific conductor configurations. The simplest are the coax and the parallel plate, which have been implemented. More complicated are rectangular or circular waveguide port sources which require fourier transforms. These have not yet been implemented. Multiple port sources may exist in the same simulation. When a port source is applied to an external surface, an absorbing boundary condition is also typically applied to the same sideset.

Other than the normal waveform description, the coaxial port requires only the spatial **center** be specified. The TEM excitation is applied over the specified **sideset** with E polarized in the radial direction and the proper $\frac{1}{r \ln \frac{b}{a}}$ variation. Here, "*a*" is the inner radius and "*b*" is the outer radius, which are determined by the code from the sideset information.

The parallel-plate port requires, in addition to the waveform description, an E-polarization **direction** and a plate separation **separ**. The TEM excitation is applied over the specified **sideset** with polarization **direction** and magnitude **scale**/**separation** such that the voltage across the plates is **scale**.

The field-distribution port obtains the port field from the sideset distribution factors in the original 3D genesis file. For a description of how to obtain this distribution, see the section on **inlet-port Poisson solutions**.

The source time history will be written to a file with a generated name in the following format: *problem_name***.prtsrc***sideset_id.proc_id***.dat**, where *sideset_id* is the **sideset** id parameter from the **port source** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, each processor who has the source is synchronized to the correct value from the owner and writes a separate source time-history file. All of these files are identical at the end of the simulation.

The source time history will be written to the hisplt database *problem*_name.**his** with the name: **COAXPORTSRC-***sideset_id* or **PPLTPORTSRC-***sideset_id.*

wire voltage source

The following syntax specifies a wire voltage source applied at a single wire node:

wire voltage source, nodeset *int***, function** *int***, scale** *real* **shift** *real*

The nodeset MUST contain only 1 node. UTDEM will issue an error if it does not.

The source time history will be written to a file with a generated name in the following format: *problem_name***.wiresrc***nodeset_id.proc_id***.dat**, where *nodeset_id* is the **nodeset** id parameter from the **wire voltage source** line and *proc_id* will be 0 for serial or the appropriate processor number for parallel. For parallel, the owning processor writes the source time-history file.

The source time history will be written to the hisplt database *problem*_name.**his** with the name: **WIREVOLSRC-***nodeset_id.*

j source

The following syntax specifies a volumetric vector current-density source:

j source, block int [*int int* **…**]**, function** *int***, scale** *real* **shift** *real*, [**direction, x** *real***, y** *real***, z** *real* **center, x** *real***, y** *real***, z** *real*, **atten** *real*, **prop** *bool* or *string*, **propdir x** *real* **y** *real* **z** *real*, **radtrans** *bool* or *string***, annulus** *bool* or *string*, **origin x** *real* **y** *real* **z** *real*]

The current density will be applied at all *nodes* in one or more mesh **block**(s), flowing in the direction **direction**, having the time history specified by **function**.

The remaining parameters are optional. If **atten** is nonzero, (default is zero) an exponential attenuation is applied across the mesh according to exp(-**atten** * z), where z is the normal distance from the source plane to the node at which the current is required. The attenuation is in the **propdir** direction, so **propdir** and **center** must be provided.

If **prop** is true or yes, (default is no) the source will emanate from a plane passing through the point **center** traveling in direction **propdir**. The source plane must be outside and behind the mesh volume with respect to the propagation direction, otherwise the code will issue a fatal error. If **prop** is false or no, the source will simply follow the specified time history simultaneously throughout the mesh block(s). If **prop** is yes, then **center** and **propdir** must be defined.

If **radtrans** is true or yes, (default is no) then the current direction will be defined by data imported from a radiation-transport code such as CEPTRE. This data is written to the simulation genesis file by the transport code.

If **annulus** is true or yes, (default is no) then a phi-directed current will be generated about the axis defined by **direction**. An **origin** for the annulus coordinate system must be provided which lies on the axis defined by **direction**.

The source time history will be written to the hisplt database *problem*_name.**his** with the name: **JSRC-1001**.

plane wave source

The following syntax specifies plane-wave source:

plane wave source, sideset *int***, block** *int* [*int int*…], **function** *int***, polarization x** *real* **y** *real* **z** *real***, propdir x** *real* **y** *real* **z** *real***,** [**center x** *real* **y** *real* **z** *real*]

A plane wave will be launched in one or more mesh **block**(s), with polarization **polarization,** propagating in the direction **direction**, having the time history specified by **function**. These **block**(s) define the total-field region which is bounded by the supplied **sideset**. Any remaining blocks in the simulation will be the scattered-field region.

The optional parameter **center** specifies the location of the phase center of the plane-wave source in mesh coordinates. This can be any point in the source plane. If this parameter is not supplied, the code will compute a phase center which is just on the incident side of the total-field region, normal to the propagation direction **propdir**. The user should take care to provide a phase-center location which makes sense relative to the total-field region and the propagation direction.

The source time history will be written to the hisplt database *problem*_name.**his** with the name: **PWSRC-***sideset_id.*

Load keywords

edge load

The following syntax specifies a single-element load on an element edge:

edge load, nodeset *int*, *type*, **value** *real*

or
edge load, edgeset *int*, *type*, **value** *real*

If **nodeset** is specified, the edge will be located which is defined by the nodeset having id **nodeset**. This nodeset should contain only 2 nodes. UTDEM will issue an error if it contains more than 2 nodes. If it contains less than 2 nodes no load will be applied. If **edgeset** is specified, the edge specified in the edgeset is used. The edgeset should contain only 1 edge. If if contains more than 1, UTDEM will issue an error. If it contains no edge, no load will be applied. The parameter *type* is limited to "R", "L", or "C": a single resistor, inductor, or capacitor, respectively. Multiple **edge load**s of the same or different *type*s may exist in the same simulation, but not on the same edge.

If it is desired to observe the voltage across the load, an observer should be assigned to the same **nodeset** using the **observer** keyword.

spice load

The following syntax specifies a load described by a Spice deck on an element edge:

spice load, nodeset *int*
or
spice load, edgeset *int*

If **nodeset** is specified, the edge will be located which is defined by the nodeset having id **nodeset**. This nodeset should contain only 2 nodes. UTDEM will issue an error if it contains more than 2 nodes. If it contains less than 2 nodes no load will be applied. If **edgeset** is specified, the edge specified in the edgeset is used. The edgeset should contain only 1 edge. If if contains more than 1, UTDEM will issue an error. If it contains no edge, no load will be applied.

Slot keyword

slot

The following syntax specifies a single sub-grid, thin-slot model:

slot, edgeset *int***, aztec_set** *int***, width** *real*, **depth** *real*,
int_mat *int*, **ext_mat** *int*

The slot is defined to lie along the edges defined by **edgeset** with the specified **width** and **depth**. Since the slot is solved using a separate linear system, an independent **aztec_set** is defined for the slot (see Linear solver

keywords). The id of this aztec_set should be something other than 0 (the default), which is reserved for the primary system solve. An example of this can be found in the input file given in Appendix II. Multiple slots may exist in the same simulation.

Since the slot algorithm requires a differential H-field drive based on fields on both sides of the PEC plane containing the slot, the materials on those sides must be defined by different material indices. These are defined by **int_mat** and **ext_mat** and are the same **material** id's as defined in the framework **block** keyword described below. These materials can in fact have identical constitutive parameters, but they must be entered as two different materials.

Wire keywords

wire

The following syntax specifies a single sub-grid, thin-wire model:

wire, edgeset *int***, aztec_set** *int***, radius** *real*

The wire is defined to lie along the edges defined by **edgeset** with the specified **radius**. Since the wire is solved using a separate linear system, an independent **aztec_set** is defined for the slot (see Linear solver keywords). The id of this aztec_set should be something other than 0, which is reserved for the primary system solve. Multiple wires may exist in the same simulation.

wire load

The following syntax specifies a single thin-wire lumped resistive load:

wire load, nodeset *int***, R, value** *real*
wire load, edgeset *int***, R, value** *real*

The resistor is defined to lie along the edge defined by the **nodeset** or **edgeset** with resistance **value**.

Hybrid keyword

hex mass lump

The following syntax specifies whether mass lumped integration is activated for hex elements:

hex mass lump, *bool* or *string*

Mass-lumped integration is required for successful hybrid simulations. The possible options are true (or yes) and false (or no). The default is presently false.

Box IEMP keyword

box iemp

The following syntax specifies a Box IEMP simulation:

box iemp [, **load current] [,load dose]**

If the **box** iemp keyword is given alone, variables are registered for DOSE and DOSE_RATE and the j source time history is normalized such that it's time integral is unity. If the **load current** keyword is added, the vector current is loaded as provided by radiation transport through the simulation genesis file. This requires the **radtrans** keyword to be specified in the **j source**. If the **load dose** keyword is added, the energy deposition is loaded as provided by radiation transport, again through the genesis file.

Note also that a **box iemp** simulation requires that the **RIC Electrical** material be used for all dielectrics. This requirement is enforced by UTDEM.

Update Mat State keyword

update mat state

The following syntax specifies whether material state is updated at the end of each time cycle:

update mat state, *bool* or *string*, **tstart** *real* **tend** *real* **interval** *real*

Options are true (or yes) and false (or no) (default). If specified, the material state is updated and a matrix refill is triggered. This is generally used only with the **Breakdown Electrical** material model. The time window over which this applies can be controlled using the **tstart** and **tend** keywords. The **interval** keyword specifies how often within the specified time window the update/refill is triggered. If **interval** is 0., then it is triggered every time cycle.

This keyword is not necessary for **box iemp** simulations using energy deposition to drive the **RIC Electrical** material model since UTDEM forces the matrix refill in that case.

Time Step Mod keyword

time step mod

The following syntax specifies a global modification to the UTDEM-computed stable time step:

time step mod, *real*

The default is 1. This value does not effect the time step if specified using the framework keyword **constant time step**.

Compute Energy Keyword

compute energy

The following syntax specifies whether or not electric, magnetic, and total energy is computed throughout the computational volume:

compute energy, *bool* or *string*

The possible options are true (or yes) and false (or no). The default is presently false. If true, appropriate energy global variables are registered and the energy time histories are compute and written to the hisplt database *problem*_name.his with the names: **E-ENERGY, H-ENERGY,** and **TOT-ENERGY.**

Framework keywords within the physics block

These keywords are really framework keywords [4] but must be placed within the physics definition keyword ("unstructured td electromagnetics") block.

block

Define a finite-element block

block *int* **material** *int*

Relates the mesh block id **block** to material definition **material**

function

Define a user-defined function for use by the **source** keyword

function *int*
time0 value0
time1 value1
time2 value2
.
.
.
end

function *int* **gaussian**, **scale** *real* **shift** *real* **width** *real*

$$scale \times \exp\left(-\left((t - shift)/width\right)^2\right)$$

function *int* **double** exponential, **scale** *real* **shift** *real* alpha *real* **beta** *real*

$$scale \times \left(\exp(-alpha \times t) - \exp(-beta \times t)\right), \quad t > 0$$

function *int* **sin squared**, **scale** *real* **shift** *real* **width** *real*

$$scale \times \sin^2\left(\frac{\pi \times t}{width}\right), \quad 0 < t < width$$

function *int* **triangle**, **scale** *real* **shift** *real* **width** *real*

$$scale \times \left(\frac{t}{width/2}\right), \quad 0 \le t < width/2$$

$$scale \times \left(1 - \frac{1}{width/2}(t - width/2)\right), \quad width/2 \le t < width$$

function *int* **sine**, **scale** *real* **shift** *real* **frequency** *real*

$$scale \times \sin\left(frequency \times t + shift\right)$$

gradual startup factor

Factor by which the initial time step is multiplied, default is 0.01

gradual startup factor *real*

Gradually increases to the initial time step. For UTDEM, the value is normally set to 1.0.

maximum time step ratio

Maximum ratio by which a time step may grow in a given cycle

maximum time step ratio *real*

Gradually increases the time step from the old to the new value.

constant time step

Specifies a constant time step for the entire simulation

constant time step *real*

If no **constant time step** is specified, UTDEM determines a time step based on the Courant stability criteria. This can provide a starting point for setting the time step. However, for the unconditionally stable **helmholtz** formulation, a much larger time step may be utilized. If this is done, the solution will remain stable but the required conjugate-gradient iterations required for system solution at each cycle will increase. Generally, for large simulation times, the overall simulation cpu time will be reduced by increasing the time step.

UTDEM PIC Input File and Keywords

The UTDEM PIC input file also uses the standard NEVADA input file format. It permits all of the UTDEM physics-specific command keywords described above, as well as several new command keywords that are described later in this section.

The keyword "**unstructured td electromagnetic pic"** specifies to the Nevada framework that the UTDEM PIC physics model should be used for the simulation. All the UTDEM keywords are available, as well as several more that are specific to UTDEM PIC. As before, case is ignored in the input file and lines are limited to 160 characters or less. And remember that when a floating-point value is required, the decimal point needs to be included (i.e., 0. not simply 0).

PIC-Specific Keywords

The following keywords are available to UTDEM PIC physics but are also available to the STDEM PIC physics model (see [5]):

define species

The **define species** command keyword group provides a means of defining charged-particle species to be used with the various particle source and diagnostic commands (currently only **beam emission** and **particle history**). One or more particle species may be defined in one **define species** keyword group.

define species
string, **mass**=real **charge state** int
…
end

Each species is specified by three required parameters. The first parameter is a string that provides a user-specified name for the species. This name is used to reference the defined species in the all other input commands that require the use of a particle species in their specification. The two remaining parameters for specifying a species are the **mass** and **charge state**, which describe the particle species mass (in AMU), and charge state, respectively. The charge state is a signed integer that gives the particle's charge relative to that of a proton. For example, an electron's charge state would be –1 and triply ionized carbon would be +3.

particle history

The **particle history** command keyword group provides a means of requesting output history diagnostics for various types of aggregate particle information. One or more such requests may be made in one **particle history** keyword group.

particle history
string, **species**=string, **status**=string
…
end

Each request has an initial string that gives the type of the request. Currently the only supported request type is COUNT, which gives the number of particles of the specified species and status. The **species** keyword allows the specification of a single particle species as defined by the **define species** command keyword group, or the special value ALL, which combines the information for all particle species in the simulation. If not explicitly specified, **species** defaults to ALL. The **status** keyword filters the aggregated information based upon the present status of the particles. Legal values for this keyword are CREATED, KILLED, and SURVIVING, which request the cumulative number of particles created during the simulation, the cumulative number of particles killed during the simulation, and the number of particles that currently survive in the simulation, respectively.

UTDEM PIC-Specific Keywords

The following keywords are available only to UTDEM PIC:

beam emission

This command keyword defines a beam emission source. The beam emission source describes a surface over which particles are emitted into the simulation region. The particle species, emission characteristics, and amplitude are provided by the several available parameter options.

beam emission, **sideset** int **species**=string
[**cycle interval** int] [**count** int] [**emit probability** real] [**spatial distribution**=FIXED|RANDOM] **energy distribution**=
CONSTANT real|RANDOM real [real]|MAXWELLIAN real
[**angle distribution**=NORMAL|CONSTANT real real|RANDOM [real real]|COSINE] [**normal tolerance** real] [AMPLITUDE real] [TEMPORAL function-set]

The **sideset** keyword is used to provide the number of the sideset in the Genesis input file that provides the set of faces that comprise the emission surface for this beam. The **species** keyword is used to provide the name of the particle species to be emitted and must match the name of a species that has been defined using the **define species** command keyword. The **cycle interval** parameter specifies the emission frequency in timesteps, with a default value of 1. **Count** specifies the number of particles emitted from each face in the emission per timestep, with a default value of 1. The **emit probability** keyword allows the specification of the probability, between 0 and 1, that any particle will actually be inserted into the system. It defaults to 1.0.

The **spatial distribution** keyword describes the spatial distribution of the emitted beam particles. Two options are available: FIXED and RANDOM. If FIXED is specified, the **count** keyword must have a value of 1, and the single particle emitted will be placed at the barycenter of the emitting face. If RANDOM is specified, the particle is placed at a random point on the emitting face. The distribution of these emission points is uniform per unit area. If not specified, **spatial distribution** defaults to FIXED, unless the **count** keyword is specified to be greater than 1, in which case it defaults to RANDOM.

The **energy distribution** keyword describes the energy distribution of the emitted beam particles. Three options are available: CONSTANT, RANDOM, and MAXWELLIAN. The CONSTANT option has a single value that gives the beam energy in electron Volts (eV). The RANDOM option has two values that give the minimum and maximum energies (in eV) for a uniform distribution of beam energy. If only one value is specified, it is assumed to be the maximum energy of the distribution, and the minimum energy is assumed to be zero. The MAXWELLIAN option has one value that gives the mean energy of the distribution in eV. For this option, the energies of the emitted particles will be distributed using

dN/dE = (1/kT) exp(-E/kT), where kT is the supplied mean energy of the distribution.

The **angle distribution** keyword describes the direction of emission with respect to the polar angle (θ) from the surface. Four options are available: NORMAL, CONSTANT, RANDOM, and COSINE. The NORMAL option has no parameters, and specifies that particles are to be emitted with an initial velocity normal to the face from which they are emitted (i.e., $\theta = 0$). The CONSTANT option has three values that represent the three components of a vector in Cartesian (x,y,z) space. Particles emitted from the emission surface will be given an initial velocity in the direction of this vector, regardless of the orientation of the emission face from which they is emitted. The RANDOM option has two values that give the minimum and maximum θ (in °) for a uniform distribution over θ. If these values are not specified, default values of 0° and 90° will be used. The COSINE option will use a cosine distribution (i.e., dN/dθ = cos θ) over the range from 0° to 90°. Note that if **angle distribution** is set to RANDOM or COSINE, the emitted particles will be randomly distributed (uniformly) in azimuth relative to the emission face. Note that if the **angle distribution** keyword is not explicitly specified, it will default to NORMAL, unless **energy distribution** is specified to be MAXWELLIAN, in which case **angle distribution** defaults to COSINE.

The **normal tolerance** keyword allows the user to specify the tolerance (ε) with which the code determines that the unit normal vectors of two distinct faces are equal. That is, if $\left|\hat{n}_1 - \hat{n}_2\right| \le \varepsilon$, they are considered to be equal. If not specified, it will default to 0.001. For all cases except **angle distribution** = CONSTANT, each face in an emission surface needs an object that contains, among other things, the face's unit normal vector in order to generate the velocity of an emitted particle. To the extent that multiple faces have the same normal vector, they can utilize the same object. Consequently, use of this keyword is primarily related to code efficiency. Note that this keyword's value provides an upper bound for the error in the normal vector for any face.

courant factor

This command keyword allows the user to specify a factor (greater than zero and less than one) to be multiplied times the maximum allowable time step.

Courant factor real

If the **constant time step** keyword (see [4]) is also specified, then the time step will be set to the smaller of the time steps determined from the two

specifications. If neither **courant factor** nor **constant time step** is specified, a Courant factor of 0.9 or 0.5 will be used by default, depending upon whether **formulation** is set to HELMHOLTZ or CURLCURL, respectively.

Note that for a PIC simulation, because of some limitations of the algorithm, there is a minimum time step Δt, related to a minimum length Δx, where $\Delta t = \Delta x/c$ and c is the speed of light. This minimum length is the minimum path length from any face of any element to any other node of that element that is not in that face. For example, for a tetrahedral element, the path for a given face is the height of the opposing node above that (triangular) face. Note that even though the Helmholtz formulation is unconditionally stable, PIC adds this constraint on the time step. For the Curl-Curl formulation, this constraint is sufficient for stability of the solver.

Framework Keywords

These keywords are framework keywords [4] which are required for a successful UTDEM simulation and appear outside the "unstructured td electromagnetics" block.

Edgeset keyword

exodus edge sets

Define specific exodus sideset(s) existing in the model which have been pre-coded by the user to be converted to edgeset(s) by the Nevada framework. This is necessary because standard ExodusII has no notion of a list of edges.

exodus edge sets (*int, int, ...***)**

Specifies a list of sideset ids existing in the Genesis file that are to be converted to Nevada edgesets. Virtual edgesets defined by a **path** keyword should not be included in this list. The sidesets on this list are the ones encoded by the preprocessor or the mesh generation software.

Material keywords

material

Define a material model or models for a material

material *int*
    model *int*
    model *int*
    …
end

Relates the material **material** to material **model**(s). For UTDEM, only one model applies to each material.

model

Define a material model

model *int string*
    [parameter real]
    [parameter real]
    …
end

Relates the material model **model** to a specific model type defined by the name *string* and defines parameters specific to that model.

For UTDEM, options for *string* and *parameter*(s) are:

    Simple Electrical
        eps *real*
        mu *real*
        sigma *real*
    end
or
    Breakdown Electrical
        eps *real*
        mu *real*
        sigma *real*
        threshold *real*
        sigma_bkdn *real*
    end
or
    HP Gas Electrical
        eps *real*
        mu *real*
        sigma0 *real*
        density *real*
        water_fraction *real*
    end

where **eps** is the relative permittivity of the medium, **mu** is the relative permeability of the medium, and **sigma** (or **sigma0)** is the conductivity of the material. The "Breakdown Electrical" model is a simple material breakdown model whereby the electric field in each element is monitored and if it reaches **threshold** V/m, the conductivity in that element is changed to **sigma_bkdn**, where it remains for the remainder of the simulation. The "HP Gas Electrical" model is a high-pressure gas chemistry model containing the additional parameters **density** and **water_fraction**.

Example input file fragments for these model specifications are shown in Fig. 3.

```
Model 1 Simple Electrical
  eps 2.
  mu 1.
  sigma 1.e-3
end

Model 2 Breakdown Electrical
  eps 2.
  mu 1.
  sigma 0.
  threshold 2000.0
  sigma_bkdn 1.e7
end

Model 3 HP Gas Electrical
  eps 2.
  mu 1.
  sigma0 0.
  density 1.23
  water_fraction 0.02
end
```

Figure 3. Typical material model descriptions.

Simulation time and output control keywords

Typical simulation time and output control keywords are shown in Fig. 4. These are described briefly below.

termination time = *real*
Total time for which to run the simulation

```
termination time = 1.e-8

emit screen, cycle interval = 1
emit plot, cycle interval = 10
emit hisplt, cycle interval = 1
plot variable
  electric_field
  econ
end
```

Figure 4. Typical simulation and output control keywords.

termination cycle = *int*
Total cycles for which to run the simulation

emit screen, cycle interval = *int*
Print status line to standard out every **cycle interval** cycles

emit screen, time interval = *float*
Print status line to standard out every **time interval**

emit plot, cycle interval = *int*
Write **plot variable**s to exodus file every **cycle interval** cycles

emit plot, time interval = *float*
Write **plot variable**s to exodus file every **time interval**

emit hisplot, cycle interval = *int*
Write global variables to hisplt file every **cycle interval** cycles

emit hisplot, time interval = *float*
Write global variables to hisplt file every **time interval**

plot variable
registered-variable name
registered-variable name
...
end

history plot variable
registered-variable name
registered-variable name
...
end
Valid plot variables for UTDEM are electric_field, magnetic_field, magnetic_flux_density, and econ (conductivity, from Simple Electrical, RIC Electrical, and Breakdown Electrical material models), and

electron_concentration, negative_ion_concentration, avalanche_rate, attachment_rate (from HP Gas material model). UTDEM_PIC adds the variables charge_density and divd_m_rho ($\nabla \cdot \overline{D} - \rho$). The valid history plot variables are the same, and for those selected, time-history data for all spatial locations specified in the **tracer points** keyword will be written to the hisplt database *problem*_name.**his.**

Linear solver keywords

Typical linear solver keywords, in this case for AZTEC [6], are shown in Fig. 5. In the first case, conjugate gradient (cg) is specified with no preconditioning but with symmetric diagonal scaling. No output is requested from AZTEC after each solve to a tolerance level of 1.e-9 with a maximum number of cg iterations set to 1000 (default is 500). The "polynomial order" should always be set to "1" for efficiency.

For the second case, cg is again specified but with jacobi preconditioning without scaling. The convergence norm is set to be relative to the rhs rather than default, which is the initial residual. If large conductivity values on the order of 1 or higher exist in the simulation, the convergence norm must be set to rhs to achieve convergence due to numerical considerations.

In the third case, the Multi-Level (ML) solver is specified to achieve convergence for very large time steps. The settings shown are typical to achieve a successful ML solution with limited testing. Others are possible and perhaps even desirable.

The fourth case mimics as closely as possible using Aztec the solver technology used in the legacy unstructured code.

Units

All UTDEM simulations are performed in MKS (SI) units. As such, all input file values must be scaled accordingly by the user. This is especially important for **box iemp** simulations since radiation-transport codes generally favor CGS units.

Running UTDEM Simulations

As mentioned previously, running a UTDEM simulation requires a Genesis mesh geometry file, *problem_name*.gen, and an input file, *problem_name*.inp (example in Appendix II). Assuming a standard Nevada user's environment or a subset thereof exists in the user's operating environment, two scripts are used to run simulations:

alegrabal –p*int problem_name*

and

Alegra *problem_name*

The **alegrabal** script invokes mesh decomposition software to divide and load balance the mesh among *int* processors. If serial execution is desired, this step is not required.

```
aztec
  solver,   cg
  precond,  none
  scaling,  sym_diag
  output,   none
  tol      = 1.e-9
  polynomial order, 1
  max iterations, 1000
end

aztec
  solver,   cg
  precond,  jacobi
  scaling,  none
  convergence norm, rhs
  output,   last
  tol      = 1.e-9
  polynomial order, 1
end

aztec
  solver,   cg
  precond,  jacobi
  output,   none
  tol      = 1.e-9
  polynomial order, 1

  multilevel
    fine sweeps = 1
    fine smoother = Hiptmair
    coarse sweeps = 6
    coarse smoother = Hiptmair
    multigrid levels = 10
    interpolation algorithm = AGGREGATION
    smooth prolongator
    hiptmair subsmoother=MLS
  end
end

aztec
  solver,   cg
  precond,  dom_decomp
  subdomain solver, icc
  type overlap, symmetric
  output,   none
  tol      = 1.e-15
  polynomial order, 1
end
```

Figure 5. Typical solver control keywords.

The **Alegra** script actually runs the simulation. For a parallel simulation being run for the first time after the **alegrabal** script is invoked, the decomposition is completed before the simulation starts. After the parallel simulation is completed, the **Alegra** script performs the necessary joining of the parallel ExodusII results into a single ExodusII file.

Inlet-port Poisson Solutions

The simulation of inlet ports where the port field distribution is derived from the static field distribution over the port is accomplished using the following process:

1. Create the full 3D mesh for the simulation, including an inlet port. The inlet must be *planar*, but can reside at an arbitrary spatial location. If meshing with I-DEAS, export this mesh to a genesis database using PREP [3]. PREP will initialize number-of-nodes-per-side distribution factors per side to zero. Other mesh generators producing generic must do the same. A suggested name for this mesh file is ***problem_name*.gen**.

2. Generate a 2D mesh description of the inlet for the Poisson solver by invoking **UTDEM Sideset Extractor** physics in Emphasis (3D). The input file for this step contains the sideset id of the inlet port to be extracted as well as the ids of any sidesets intersecting the extracted sideset. These intersecting sidesets are those required to set boundary conditions for the 2D Poisson solve. The sideset extractor will extract the inlet sideset, convert it to a 2D mesh description, and write a 2D gen file for Emphasis. The extractor will also record in this gen file the appropriate transformation matrix to be applied to the Poisson results to properly position the Poisson solution into the 3D gen file. An example input file is shown in Appendix III. The extractor will create the 2D mesh file using the name ***problem_name*.2D.gen**.

3. Solve the Poisson problem by invoking **CABANA POISSON** physics with the **Poisson solution** keyword in Emphasis (2D). After the solution is obtained, Emphasis will write the solution and the transformation matrix into the 3D gen file for UTDEM. For this step, a new *copy* of the full 3D gen file created in step 1 must be available in the directory so that the Poisson results can be written into it. If the 3D gen file has already been modified in this manner, the Poisson solution will note this and ask the user to provide a new copy. Since this is a generated file, a suggested name for this copy is ***problem_name*.inlet.gen**.

   An example input file is shown in Appendix IV. Here, the **Poisson solution** keyword specifies that Cabana obtain a single Poisson solution on the mesh with the given boundary conditions and exit. In addition, a constant **charge density** can be supplied for the mesh volume. An ascii file will also be written containing the results with filename **results file**. Dirichlet boundary conditions are supplied with the **conductor** keyword. The **export results** keyword controls the writing of

results to the UTDEM 3D **genesis file** and **sideset** specified. Further information can be found in [7].

4. Invoke **UTDEM** physics in Emphasis (3D) to complete the full 3D solution, specifying *field dist* in the port source keyword descriptor.

Conclusion

This document, along with the user guides for a modeling and meshing tool such as I-DEAS should allow the user to successfully utilize EMPHASIS/Nevada UTDEM to simulate a wide variety of EM effects. To gain experience, the UTDEM regression suite contains several simulations exercising most if not all of the code options. These should be examined and well understood before attempting more involved simulations.

References

1. C. D. Turner, *Emphasis/NEVADA Unstructured FEM Implementation*, Sandia National Laboratories Report SAND2003-0847, Nov. 2003.

2. L. A. Schoof, V. R. Yarberry, EXODUSII: A Finite Element Data Model, Sandia National Laboratories Report SAND92-2137, reprinted Sep. 1996.

3. M. F. Pasik, et.al., *VOLMAX User's Guide*, unpublished but available here: VolMax User's Guide.

4. S. K. Carroll, et. al., *ALEGRA: Version 4.6,* Sandia National Laboratories Report SAND2004-6541, Jan. 2005.

5. R. S. Coats, M. F. Pasik, and D. B. Seidel, *Emphasis/Nevada STDEM User's Guide Version 1.0*, Sandia National Laboratories Report, under preparation, Jan. 2004.

6. R. S. Tuminaro, M. Heroux, S. A. Hutchinson, J. N. Shadid, *Official Aztec User's Guide Version 2.1*, Sandia National Laboratories Report SAND99-8801J, Nov. 1999.

7. C. D. Turner, CABANA User Guide Version 1.0, Sandia National Laboratories Report SAND2005-XXXX, March 2005.

Appendix I. Use of the PREP Mesh Preprocessor

The mesh preprocessor PREP is a holdover from legacy code that has been refactored to provide conversion from I-DEAS Universal format to ExodusII format for EMPHASIS/Nevada UTDEM. It is also used to provide encoded sidesets for subsequent conversion to edgesets by Nevada. Use of PREP is not required if 1) the chosen mesh-generation software creates ExodusII format directly including all required nodesets, sideset-coded edgesets, and sidesets, or 2) the simulation requires no edgesets (or only virtual edgesets) but otherwise the mesh-generation software can provide the required ExodusII description including nodesets and sidesets.

Reference [3] provides general guidance on the use of PREP but a few additional comments are warranted. A typical PREP input file for a pure unstructured UTDEM simulation is shown below.

```
$INPUT
flagwrap   =  'N'
flagblocks =  'N'
flagchaco  =  'Y'
flagalegra =  'Y'
nodeblu    =  'HLinInt1'
nodegryblu =  'ELinInt1 HLinInt1'
nodeltblu  =  'ELinInt2 HLinInt1'
nodemag    =  'ELinInt1'
nodepnk    =  'ELinInt2'
nodecyn    =  'Load1'
nodegrn    =  'Load2'
nodeorg    =  'Source1'
nodeltmag  =  'Obs'
$END
```

The “flagwrap”, “flagblocks”, “flagchaco”, and “flagalegra” keywords must be set as shown for pure unstructured. If this were a hybrid FDTD/FETD simulation, “flagwrap” and “flagblocks” would be set to ‘Y’.

The “node*” node color -> attribute keywords are required to assist PREP in making connections between our I-DEAS meshing convention of using node color to represent attributes requiring nodesets or edgesets, including non-port sources, observers including slot and wire observers, loads, slots, and wires. Attributes requiring mesh-related edgesets, such as ElinInt (E Line Integral) and wire, also require beam elements along the path to further assist PREP in correctly defining the edgeset. Attributes utilizing sidesets and virtual edgesets do not appear in the PREP input file. Sidesets are generated within I-DEAS by utilizing “pressure” or “force” boundary conditions.

Node color keywords containing more than one attribute, such as “nodegryblu” above, indicate that those paths intersect in the mesh. In this case, an E Line Integral path intersects an H Line Integral path. Therefore, both attributes must be assigned to that node color so PREP can correctly include that node in both paths.

Appendix II. Complete UTDEM input file

```
$-----------------------------------BEGIN_QA---------------------------
$ ID:         ucavabc_slots
$ Title:      Cavity w/slots surrounded by ABC
$ Category:   Regression
$ Physics:    electromagnetics
$ Dimension:  3D
$ Owner:      C. David Turner
$
$ Description:
$
$   Conducting cavity with internal source and slots in walls.
$   Energy leaks out through slots to observer outside cavity.
$   ABC surrounds entire object.
$
$-----------------------------------END_QA-----------------------------
TITLE
  Unstructured 3D cavity with edge source and observer

$
$ The following line will have nevada print out the sequence of calls
$ it makes during execution.
$
$DEBUG MODE, LOCATION
$ The following line dumps out node/edge/face/element connectivity
info.
$DUMP FACES

UNSTRUCTURED TD ELECTROMAGNETICS
  formulation, helmholtz
  aztec set, 0
  abc bc, sideset 4
  pec bc, sideset 2
  observer, nodeset 28
  observer, nodeset 29
  source, nodeset 31, function 1
  slot observer, nodeset 19
  slot, edgeset 123, aztec_set 1, width 0.00001, depth 0.0, int_mat 1,
ext_mat 2
  slot observer, nodeset 20
  slot, edgeset 124, aztec_set 2, width 0.00005, depth 0.0, int_mat 1,
ext_mat 2

  CONSTANT TIME STEP 1.01197539e-09
  GRADUAL STARTUP FACTOR 1.0

  BLOCK 1
    MATERIAL 1
  END
  BLOCK 22
    MATERIAL 2
  END
END
FUNCTION 1 GAUSSIAN, SCALE=1.0 WIDTH=2.0e-9 SHIFT=10.0e-9
```

```
EXODUS EDGE SETS (123 124 153)

$$$$$$$$$$$$$$$$$$$ execution control $$$$$$$$$$$$$$$$$$$

TERMINATION TIME 9.0e-9

$$$$$$$$$$$$$$$$$$$ solver control    $$$$$$$$$$$$$$$$$$$

aztec
  solver,    cg
  precond,   jacobi
  output,    none
  tol        = 1.e-12
  polynomial order,  1
end

aztec 1
  solver,    cg
  precond,   jacobi
  output,    none
  tol        = 1.e-12
  polynomial order,  1
end

aztec 2
  solver,    cg
  precond,   jacobi
  output,    none
  tol        = 1.e-12
  polynomial order,  1
end

units, si

$$$$$$$$$$$$$$$$$$$ output control    $$$$$$$$$$$$$$$$$$$

EMIT SCREEN, CYCLE INTERVAL = 1
EMIT PLOT, CYCLE INTERVAL = 2
PLOT VARIABLE
  ELECTRIC_FIELD
END

$$$$$$$$$$$$$$$$$$$ material models $$$$$$$$$$$$$$$$$$$

MATERIAL 1
  model 1
END

MATERIAL 2
  model 2
END

MODEL 1 SIMPLE ELECTRICAL
  EPS 1.
  MU 1.
  SIGMA 0.
```

```
END

MODEL 2 SIMPLE ELECTRICAL
  EPS 1.
  MU 1.
  SIGMA 0.
END

EXIT
```

Appendix III. Sideset Extractor input file

```
$-----------------------------------BEGIN_QA---------------------------
$ CVS: $Id$
$-----------------------------------END_QA-----------------------------

$debug mode, LOCATION

title
  SIDESET extract: extract a side set and turn it into a 2D mesh file

$$$$$$$$$$$$$$$$$$$ physics options $$$$$$$$$$$$$$$$$$$

UTDEM Sideset Extractor
  extract, sideset 10
  intersect, sideset 9, sideset 14, sideset 15, sideset 16, end

  block 1
    material 1
  end

end

$$$$$$$$$$$$$$$$$$$ execution control $$$$$$$$$$$$$$$$$$$

termination cycle = 1

emit plot, cycle interval = 1

$$$$$$$$$$$$$$$$$$$ material models $$$$$$$$$$$$$$$$$$$

Material 1
end

exit
```

Appendix IV. Poisson Solution input file

```
$-----------------------------------BEGIN_QA---------------------------
$ CVS: $Id$
$-----------------------------------END_QA-----------------------------

$debug mode, LOCATION
$debug mode, CABANA

title
  CABANA: Poisson solution

$$$$$$$$$$$$$$$$$$$ physics options $$$$$$$$$$$$$$$$$$$

CABANA POISSON

  Poisson solution, charge density 0., results file "inlet.out"

  conductor, sideset 9, potential 0.
  conductor, sideset 14, potential 0.
  conductor, sideset 15, potential 1.
  conductor, sideset 16, potential 1.

  export results, genesis file "z_vert.inlet.gen", sideset 10

  block 10
    material 1
  end

end

double precision exodus

aztec
  solver,   cg
  precond,  none
  scaling,  sym_diag
  output,   none
  tol      = 1.e-6
  polynomial order, 1
end

units, si

$$$$$$$$$$$$$$$$$$$ execution control $$$$$$$$$$$$$$$$$$$
$ Nevada requires termination time to be specified:
termination time = 1.e-8

$$$$$$$$$$$$$$$$$$$ material models $$$$$$$$$$$$$$$$$$$

emit screen, cycle interval = 1
emit plot, cycle interval = 1

plot variable
  potential
```

```
  electric_field
  charge_density
end

$$$$$$$$$$$$$$$$$$$$ material models $$$$$$$$$$$$$$$$$$$$

Material 1
  Model 1
end

Model 1 Simple Electrical
  eps 1.
  mu 1.
  sigma 0.
end

crt: off
exit
```

Distribution

1	MS1152	M. L. Kiefer, 01642
1	MS1152	L. I. Basilio, 01642
1	MS1152	R. S. Coats, 01642
1	MS1152	J. D. Kotulski, 01642
1	MS1152	M. F. Pasik, 01642
1	MS1152	T. D. Pointon, 01642
1	MS1152	D. B. Seidel, 01642
1	MS1152	C. D. Turner, 01642
1	MS0378	W. J. Bohnhoff, 09231
1	MS9018	Central Technical File, 8945-1
2	MS0899	Technical Library, 9616